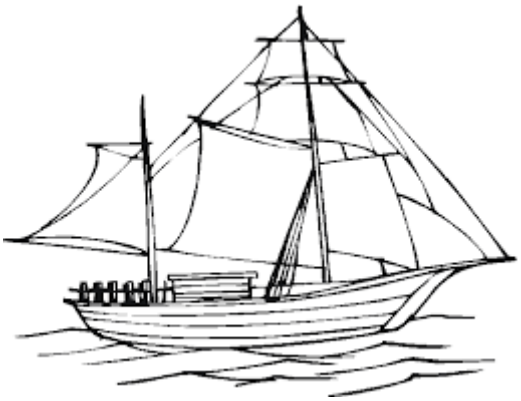
UV3.6 Système d'exploitation

Projet Simulateur



Préparé par :

TANIOS Georges

SAAD Ali

ALSAKAIT Mohammed

Professeur :

LAGADEC Loïc

Table des matières

Introduction :

Le but du projet est de développer un système client-serveur ou le serveur gère la navigation des bateaux qui sont considères comme des clients dans notre système.

Déroulement du projet :

Dans ce projet nous développons deux grandes parties client et serveur. Le serveur envoie toutes les informations qui concernent le vent et les coordonnées des rochers à tous les autres bateaux.

Une autre tâche qu'il faut faire, est de modélise l'environnement maritime « océan » qui est caractérisé une dimension finie (x,y). Dans l'océan il y a des rochers qu'ils sont évités par les bateaux. Et les bateaux avance en prenant en compte la vitesse et la direction de vent.

Serveur:

Nous avons créé le fichier serveur, nous l'avons donné une porte fixe « 4444 ». Le type du serveur que nous avons utilisé est TCP, l'adresse est l'IP local de la machine « 127.0.0.1 », puis nous avons changé dans notre code pour rendre le serveur multi client dont nous avons fait une boucle sur la fonction accepte de connexion.

Le serveur alors lit les coordonnées des rochers du fichier « carte.txt » et l'envoyé à chaque client connecté au serveur, la même pour la vitesse et la direction du vent sont entré par l'utilisateur dans le serveur en ligne de commande, après ils sont envoyés au client connecté. Mais nous pourrons varier les paramètres du vent au niveau de la console. L'utilisateur passera la variation sous forme d'une ligne de commande au serveur qui est en écoute permanente.

Alors la communication est bonne entre le serveur et les clients.

Client :

Ici dans cette deuxième, partie nous avons créé un fichier client considérer comme un bateau. Au démarrage de fichier client « bateau » nous mettons en ligne de commande le nom, la vitesse, la direction du bateau et encore ces coordonnées initial.

Pour rendre la démarche du bateau au cas réel. Nous prenons en considération la direction et la vitesse de vent, Et bien sûr que le bateau doit éviter les rocher dans l'océan, et éviter encore les bateaux voisins.

Nous avons créé un algorithme pour éviter les rochers dans l'océan.

La méthode de calculer la vitesse est celons la direction et la vitesse du vent d'une cote a la direction et la vitesse du bateau d'une autre cotes :

. Lorsque la direction du vent et du bateau sont similaires, la vitesse du bateau sera augmentée et suivra la règle :

$$vitesse\ finale = vitesse\ initiale * (1 + force\ du\ vent/10)$$

. Sinon elle sera réduite et calculée de la manière suivante :

$$vitesse\ finale = vitesse\ initiale * (1/force\ du\ vent)$$

. Lorsque le vent est perpendiculaire à la direction prise par le bateau, il se déplacera à la vitesse calculée selon la méthode ci-dessus mais en diagonale soit un décalage d'une case dans le sens du vent.

Le client possède un menu :

- ➔ 1. Modification de la vitesse du bateau.
- ➔ 2. Modification de la direction du bateau.
- ➔ 3. Visualiser l'océan.
- ➔ 4. Quitter.

Ici c'est un exemple de changement du vent

```
saadal@P32042:~/Uni/SE/projet/10-1-2019$ gcc -o c1 clientTCP.c
saadal@P32042:~/Uni/SE/projet/10-1-2019$ ./c1 bateau1 5 N 10 12
le nom est bateau1
la vitesse est 5
la direction est N
X est 10
Y  est 12
Programme Client TCP
Appel Fonction Socket
Demande de connexion
Connxion etablie attente

Attente si il ya un changement du vent

vent : vitesse 0 et la direction:N

Attente si il ya un changement du vent

vent : vitesse 2 et la direction:S

Attente si il ya un changement du vent
```

```
 donne moi la force et la direction du vent: ^C
saadal@P32042:~/Uni/SE/projet/10-1-2019$ gcc -o s tcpServer.c
saadal@P32042:~/Uni/SE/projet/10-1-2019$ ./s
[+]Server Socket is created.
[+]Bind to port 4444
[+]Listening....
Connection accepted from 0.0.0.0:0

 donne moi la force et la direction du vent: 0N

 donne moi la force et la direction du vent: 2S

 donne moi la force et la direction du vent:
```

Ici l'exemple de la modification de la vitesse du bateau.

```
saadal@P32042:~/Uni/SE/projet/10-1-2019$ gcc -o c1 clientTCP.c
saadal@P32042:~/Uni/SE/projet/10-1-2019$ ./c1 bateau1 5 N 10 12
le nom est bateau1
la vitesse est 5
la direction est N
( est 10
/  est 12
Programme Client TCP
Appel Fonction Socket
Demande de connexion
Connxion etablie attente

Attente si il ya un changement du vent



**********************************************************************
**********************  MENU  ********************************
**********************************************************************
 1) Modification de la vitesse du bateaux
 2) Modification de la direction du bateaux
 3) visualisation de l ocean
 4) QUIT
 donner votre choix:1

 changer la vitesse du bateaux :5

 la nouvelle vitesse est: 5

**********************************************************************
**********************  MENU  ********************************
**********************************************************************
 1) Modification de la vitesse du bateaux
 2) Modification de la direction du bateaux
 3) visualisation de l ocean
 4) QUIT
 donner votre choix:
```

Les fonctions :

- Premièrement nous avons faire une connexion entre les clients et le serveur utilisant la méthode de communication « socket ».

- Au démarrage du client nous ajoutons en ligne de commande les paramètres du bateaux « nom, vitesse, direction, x, y ». Ensuite toutes ces paramètres sont enregistrer dans un fichier appeler « bateaux.txt ».

Voici l'exemple :

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
saadal@P32042:~/Uni/SE/projet/10-1-2019$ gcc -o c1 clientTCP.c
saadal@P32042:~/Uni/SE/projet/10-1-2019$ ./c1 bateau1 5 N 10 12
le non est bateau1
la vitesse est 5
la direction est N
X est 10
Y  est 12
Programme Client TCP
Appel Fonction Socket
Demande de connexion
Connxion etablie attente

Attente si il ya un changement du vent
```

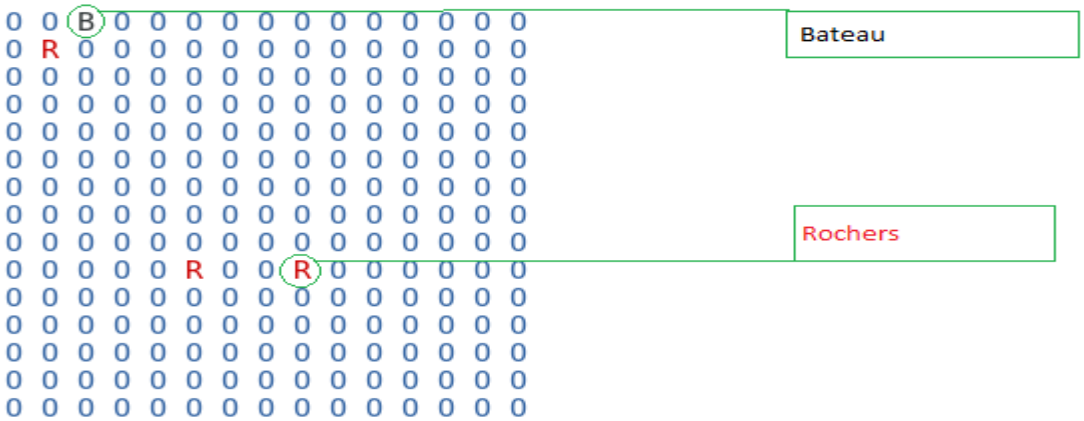
- Le client lit les coordonnée des rochers, après il les stocke dans une variable.

- Toutes les informations concernant les rochers et les bateaux seront dans des « array » dans notre code.

- Notre Système est constitué de deux threads qui marchent en parallèle :
  - Thread1 : Responsable pout l'affichage de menu et ces fonctions et gère le mouvement des bateaux.
  - Thread2 : Le thread 2 est utilisé pour que notre client reste en attente permanent de la force et de la direction du vent entrer par le serveur.

- Nous avons pris en considération la vitesse du vent qui affecte sur la vitesse final des bateaux, nous avons utilisé les formules déjà mentionne dans la premier partie du rapport.

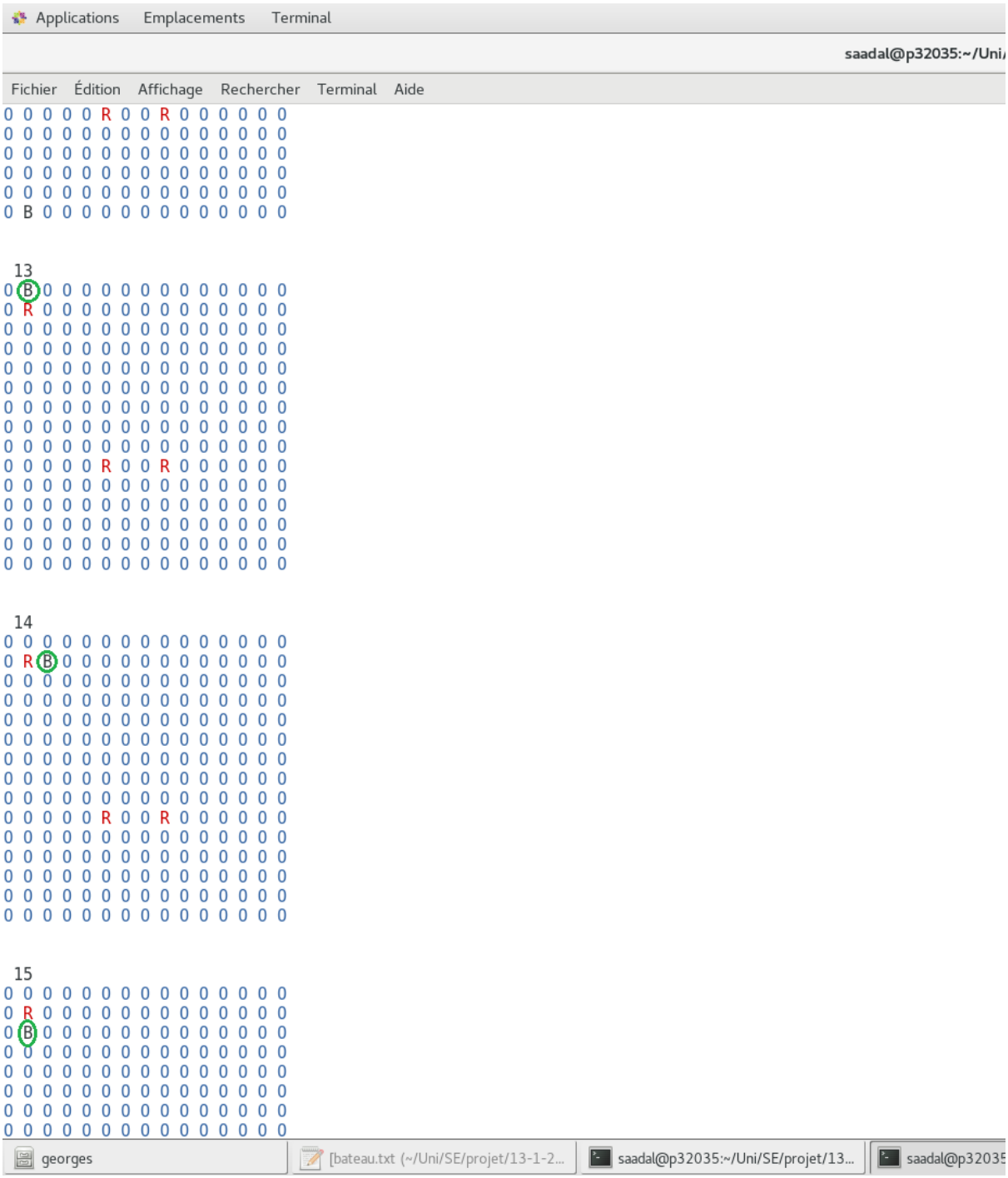
- Pour l'affichage on a utilisé les matrices que nous avons stocké toutes les informations concernant l'océan comme rocher et bateaux. Le type du matrices est une chaine de caractère double-dimensions où « R » Désigne les rochers, les bateaux sont Désignés par la lettre « B ». Si non il y a« O » qui désigne l'océan.



- Pour visualiser les déplacements des bateaux il faut choisir du menu le numéro «4 « ce qui est la « visualisation ».

- Chaque bateau qui se déplace dans l'océan et évite tous les rochers devant elle et tous les bateaux encore. Quand le bateau arrive à la fin de l'océan, il se ramène tous seule au début de l'océan.

  Comme vous voyez dans cette photo que le bateaux évite le rocher devant lui pendant l'étape 14.



Pour le lancement du client : gcc -o client clientTCP.c -lpthread

Pour le lancement du serveur: gcc -o serveur tcpServer.c

Conclusion :

Le but de ce projet était de pouvoir se communiquer entre un serveur et du client. Et d'échanger des informations entre eux pour arriver à bouger les bateaux en utilisant plusieurs mécanismes que nous avion déjà appris dans le TP comme le socket les threads etc….